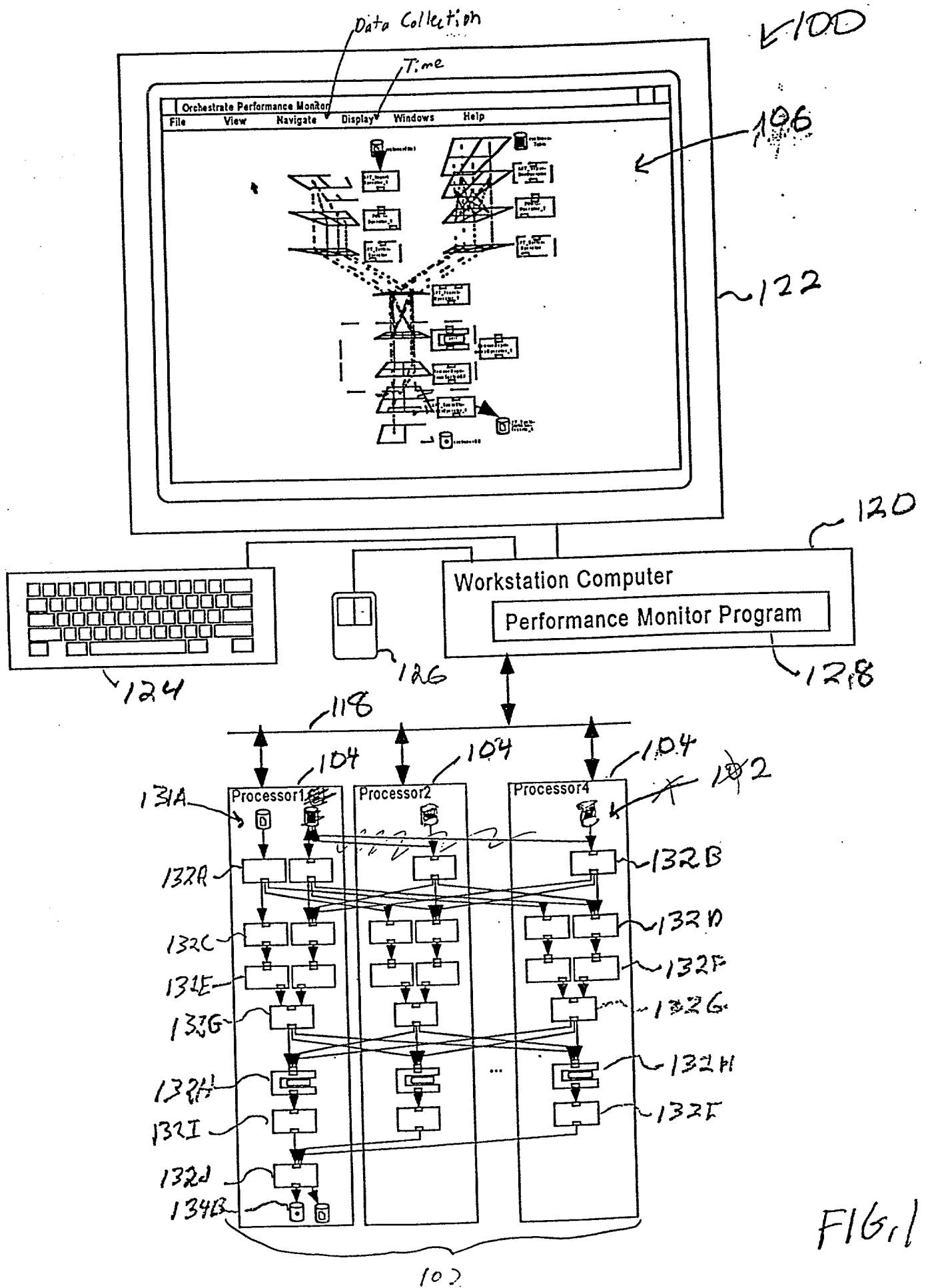
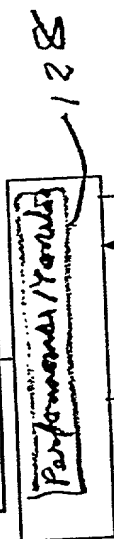
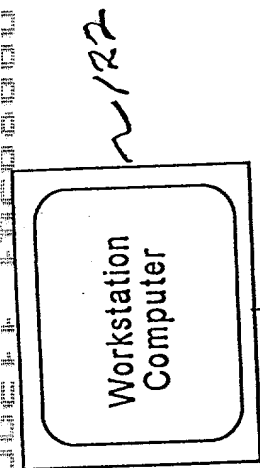


FIG. 100



100

FIG. 1



120

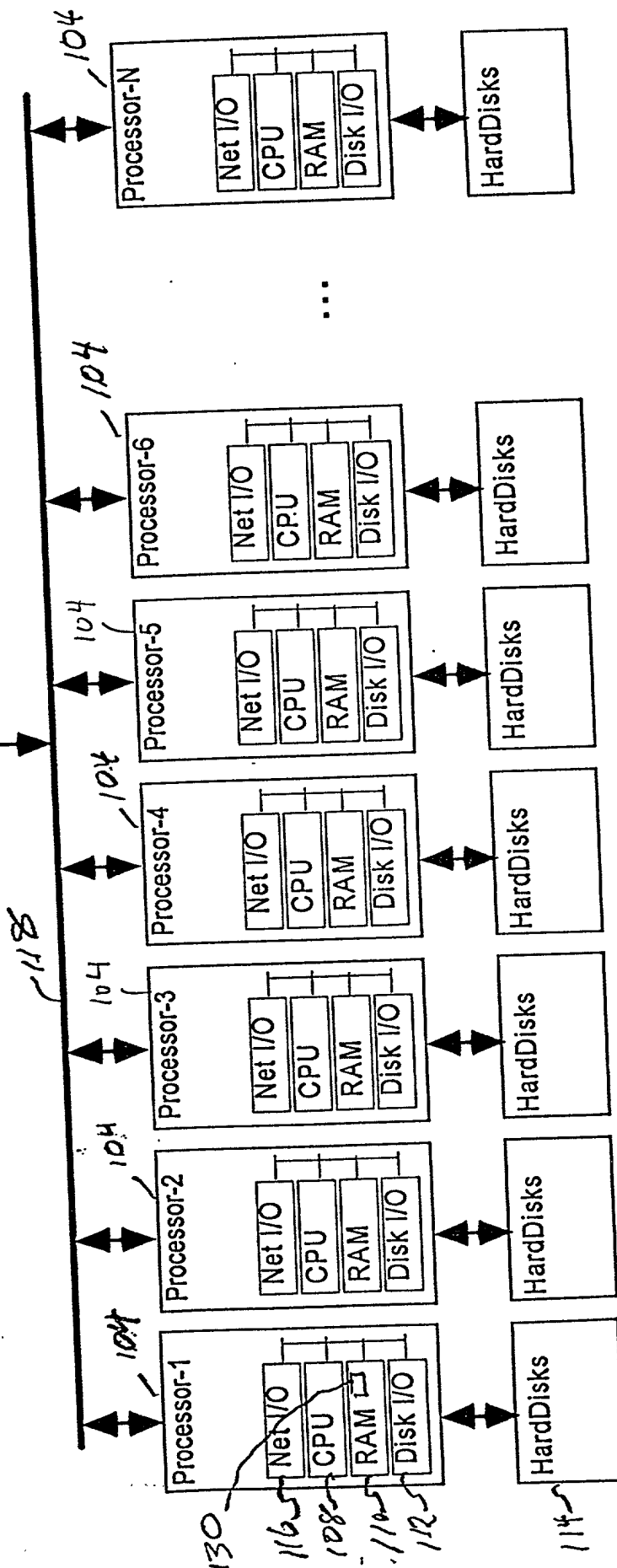


FIG. 2

FIG. 3

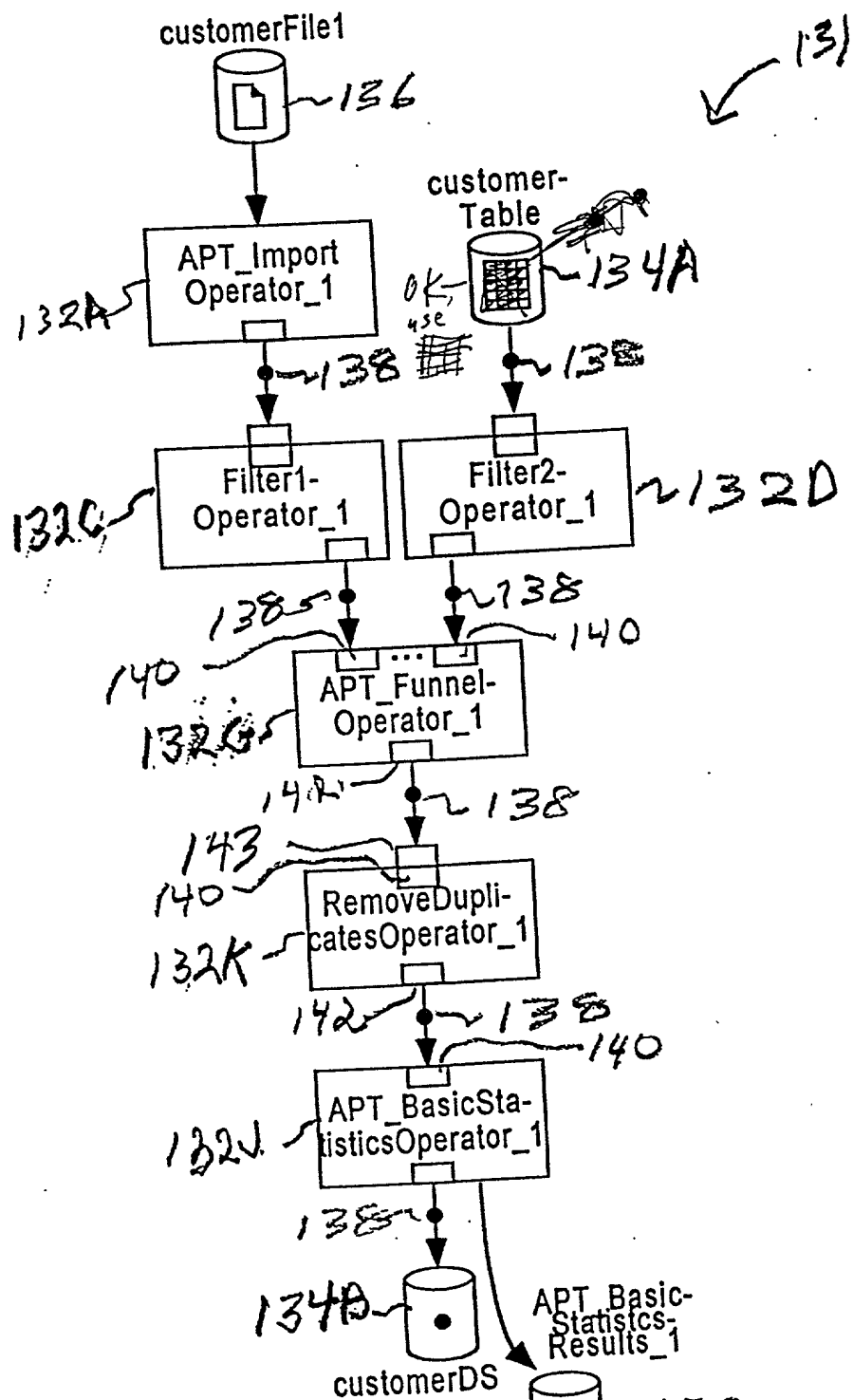


FIG. 3

1322

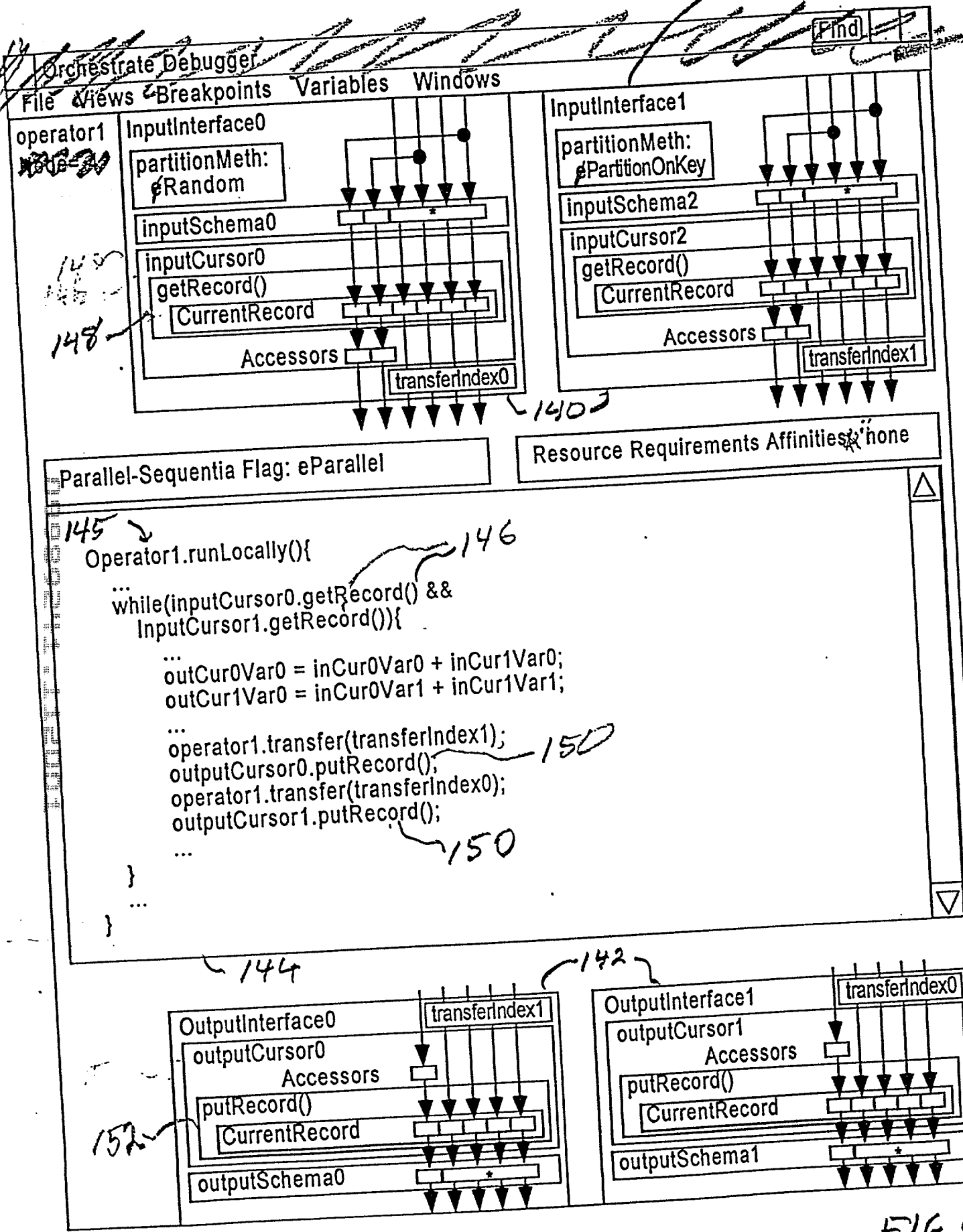


FIG. 4.

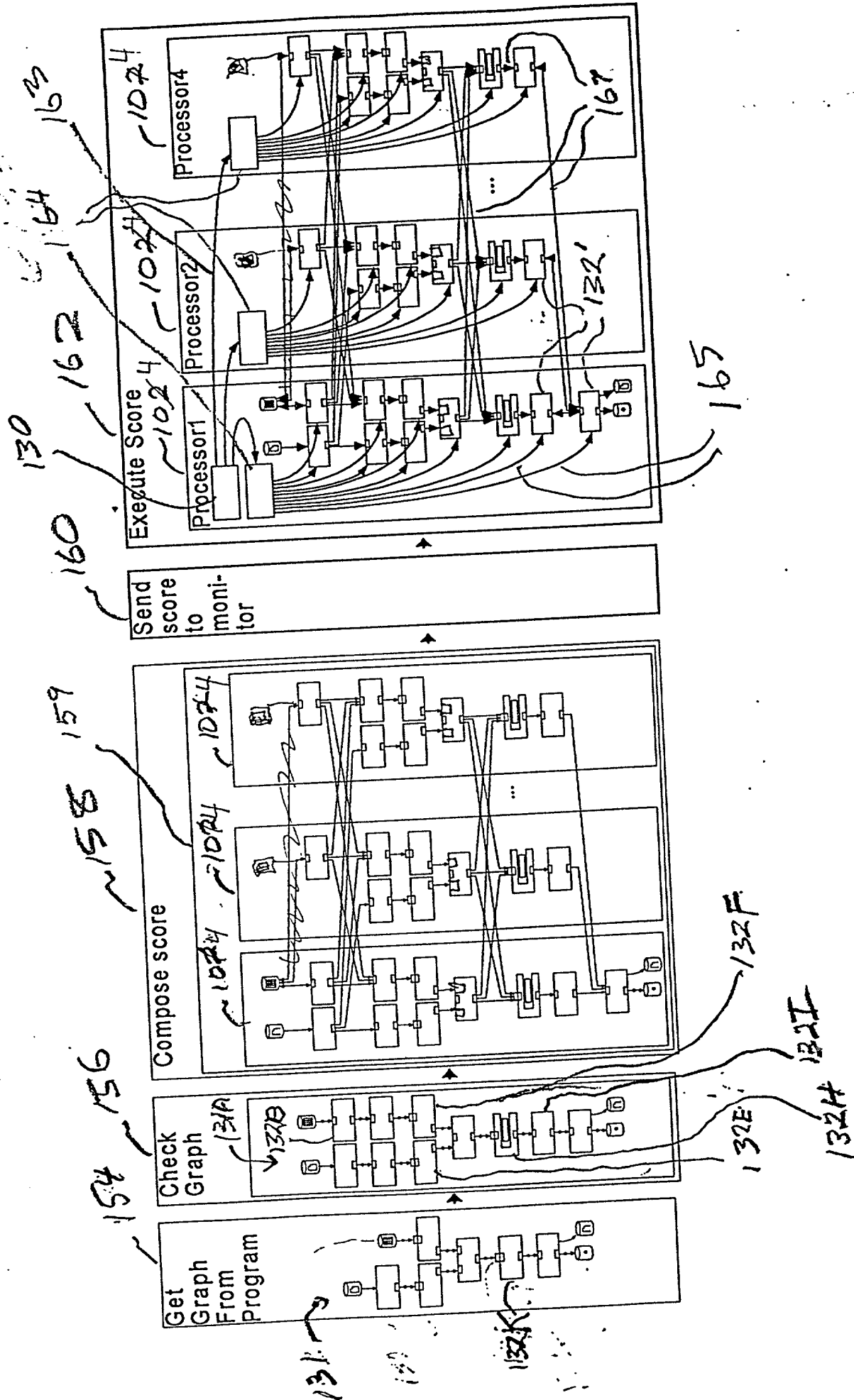


FIG. 5

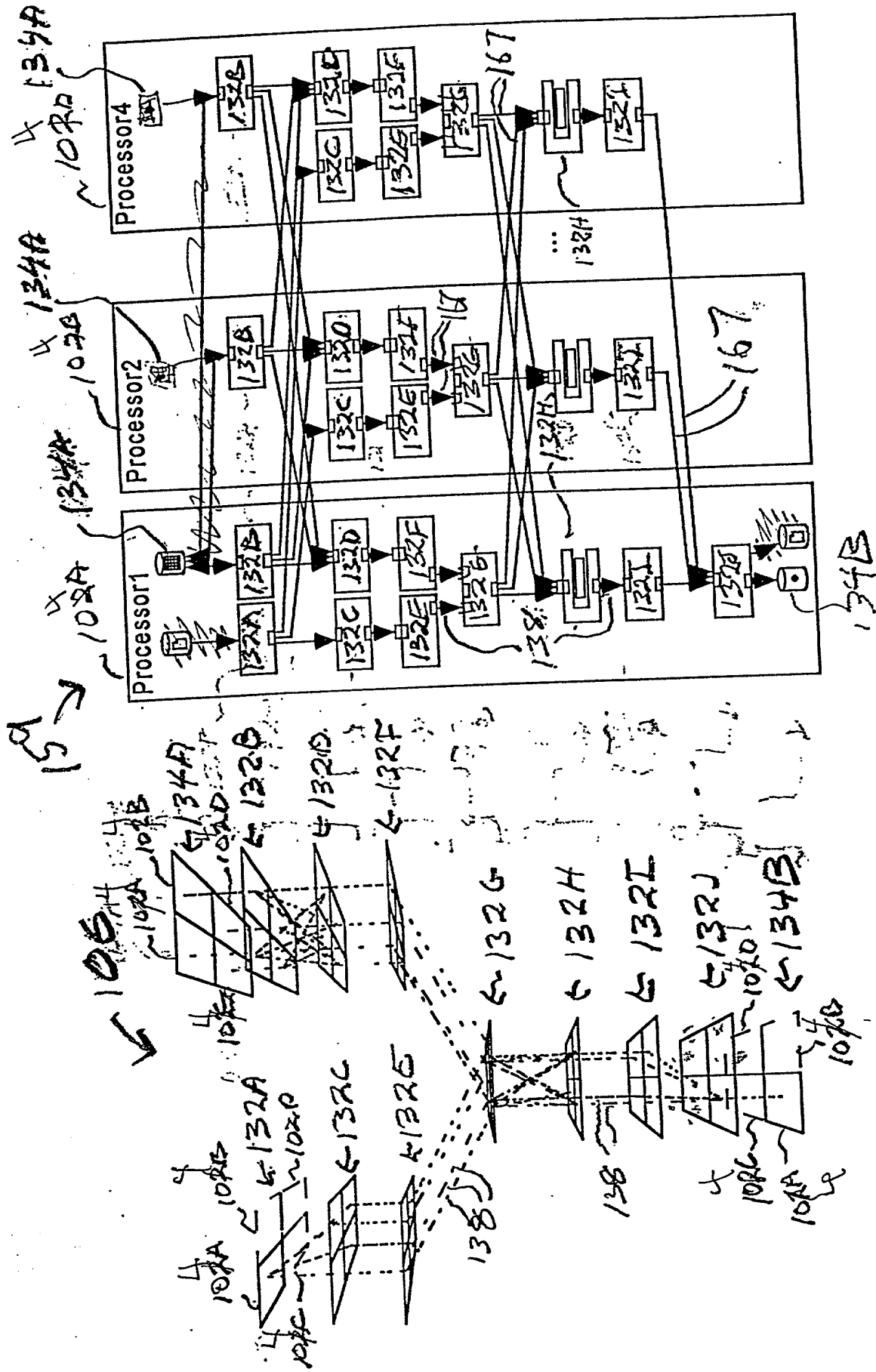


FIG. 6

Foot print collection 302

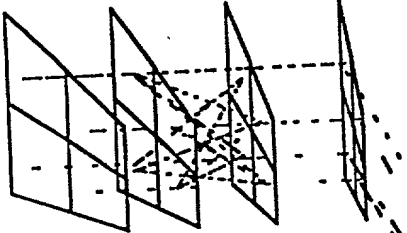
✓ 171

✓ 170

Orchestrate Performance Monitor

File View Navigate Display THE Windows Help

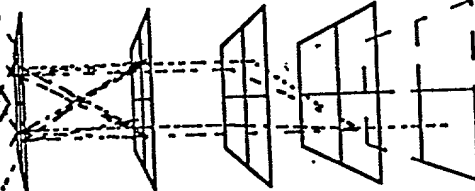
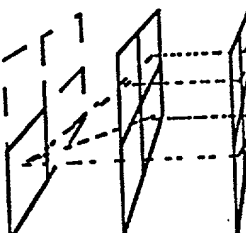
5106



276

174

278 199



✓ 172

F167

176

182

178

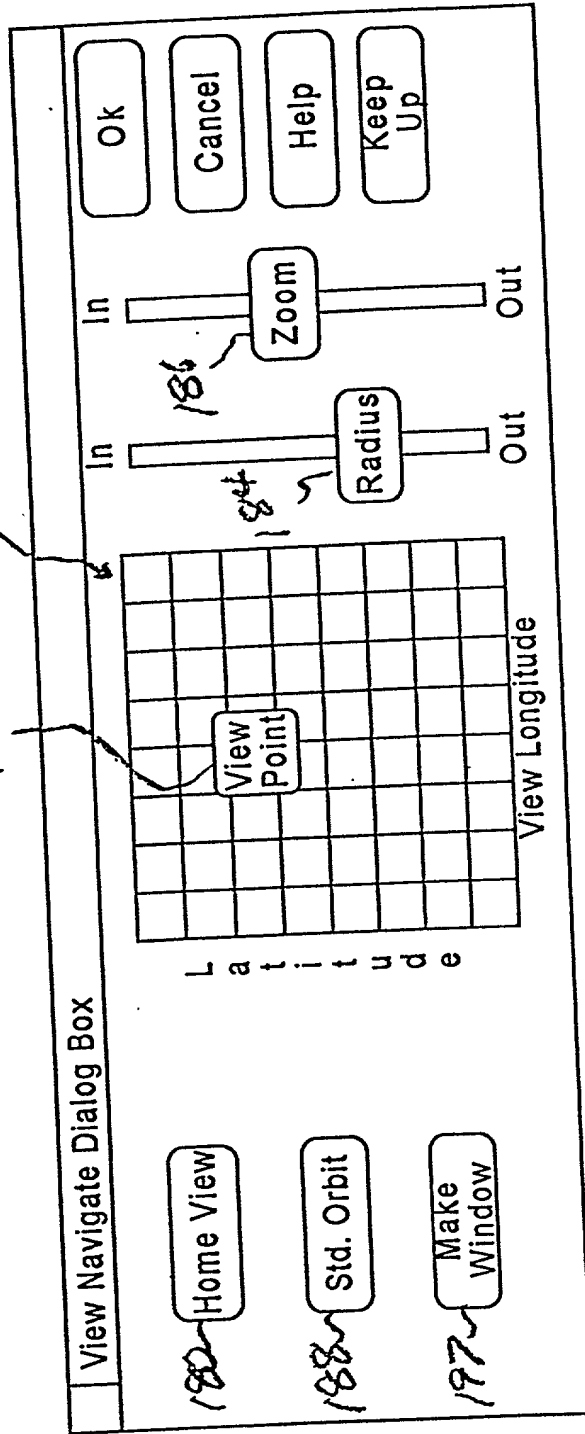


FIG. 8



- View Menu~198
  - 2D Views~200
    - By Instances~216
    - By Operators~218
    - ...
  - 3D Views~202
    - Expansion Level~220
      - By Instances~228
      - By Operators~230
      - By Levels~232
      - By Processors~234
      - By System~236
    - ...
  - Display Ports~203
  - Datalink Display~204
  - Labels~205
    - No Labels~~~268~~ 254
    - Name Only~~~270~~ 255
    - Block Diagram~272
  - Bar Graphs~207
  - Number Processors~206
  - Get & Put Hangs~210
  - EOF Display~211
  - Monitored field~212
  - Saved Visualization Manager~213

Fig. 9



# Data Collection

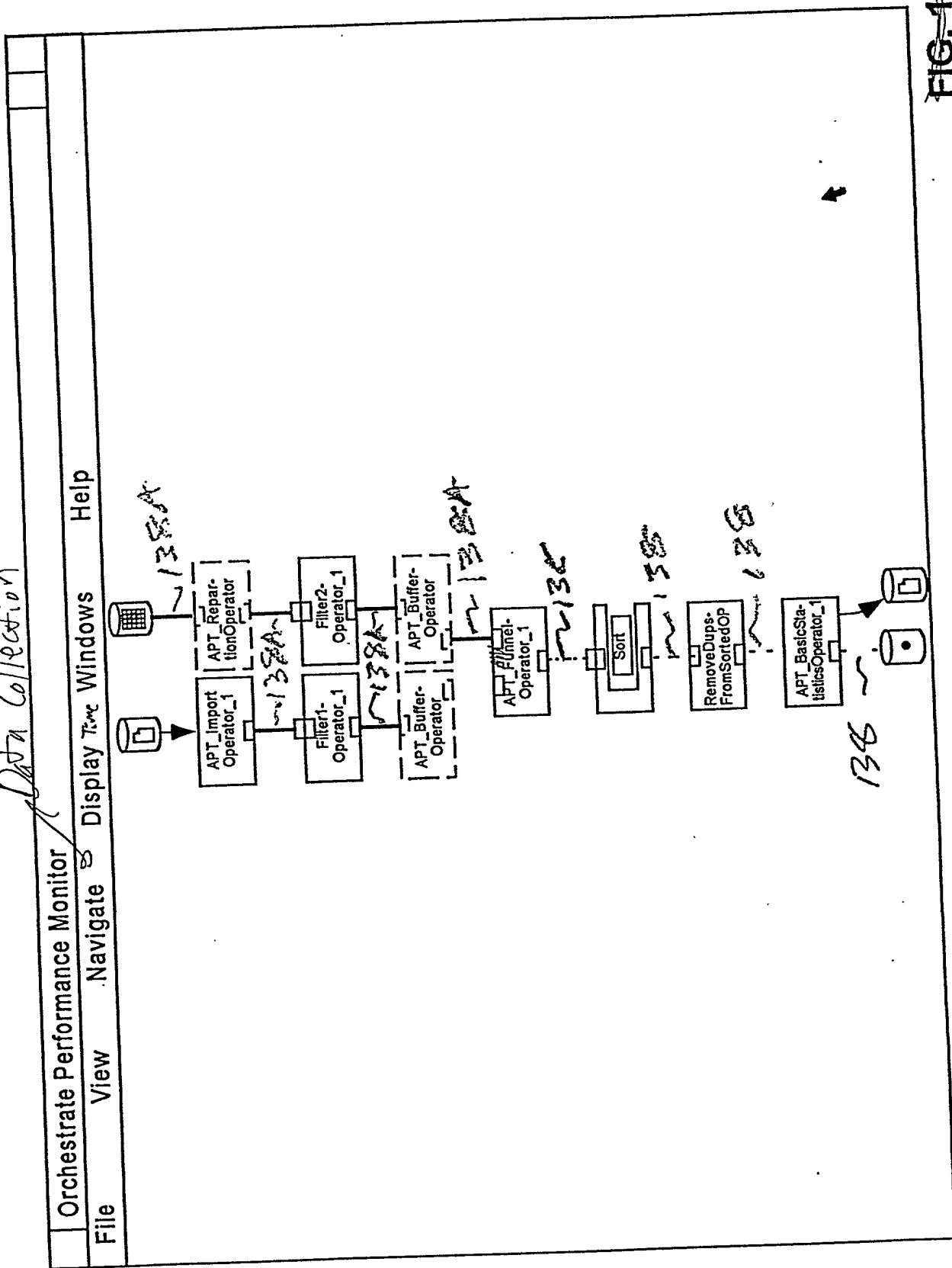


FIG. 4  
FIG. 11

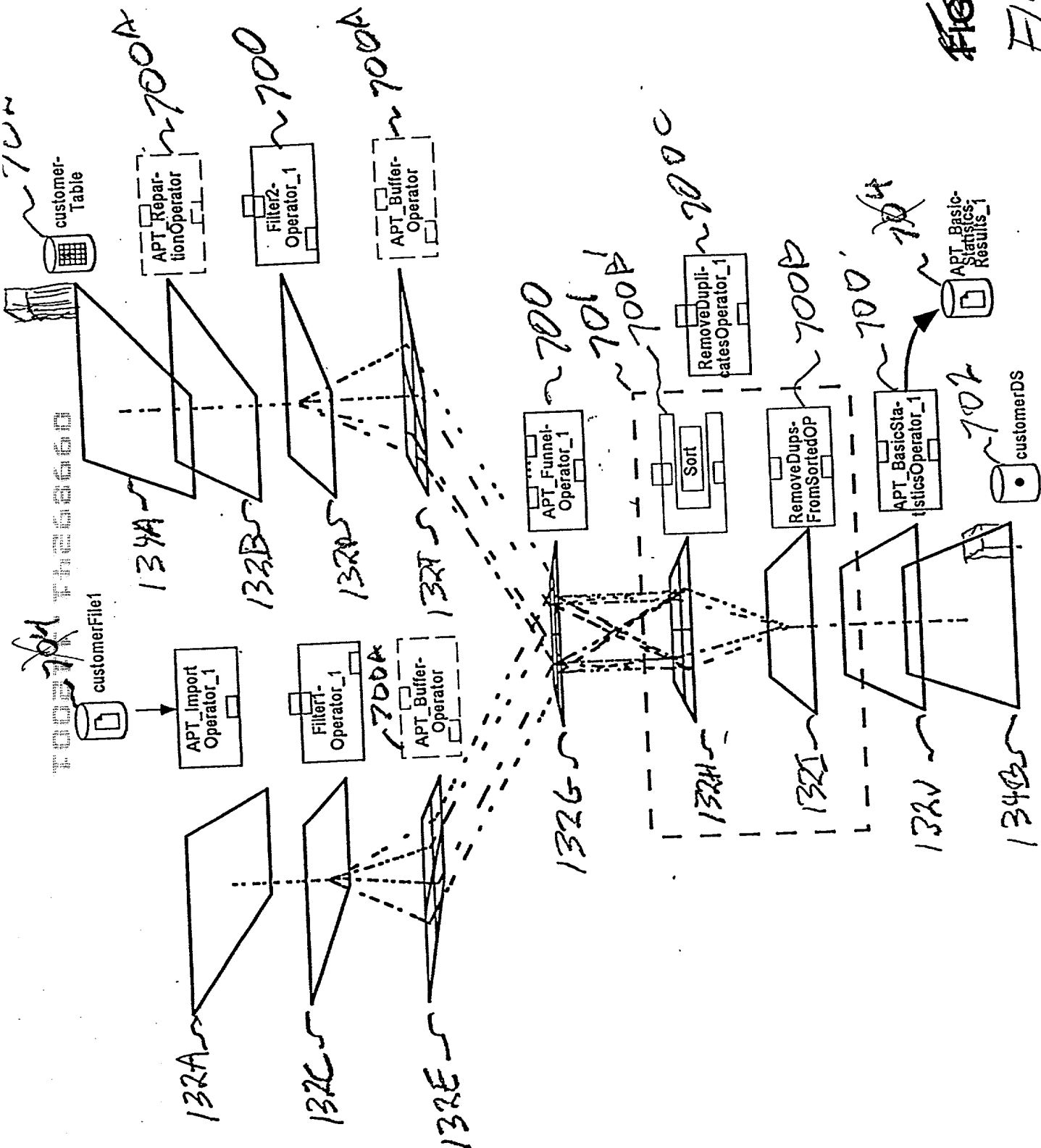
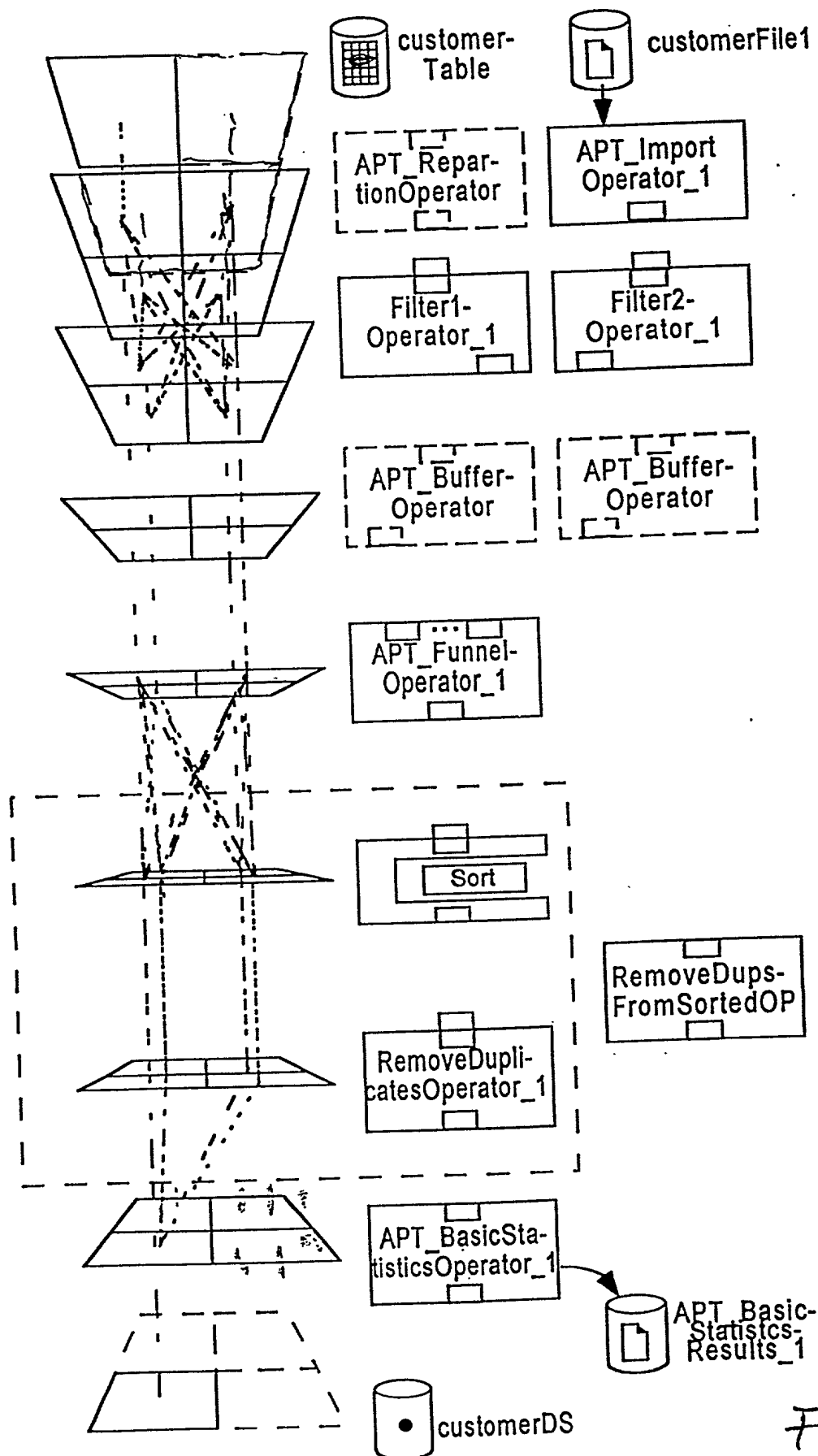


FIG. 12  
FIG. 12

Figure 1: APT Data Processing Flow



71613

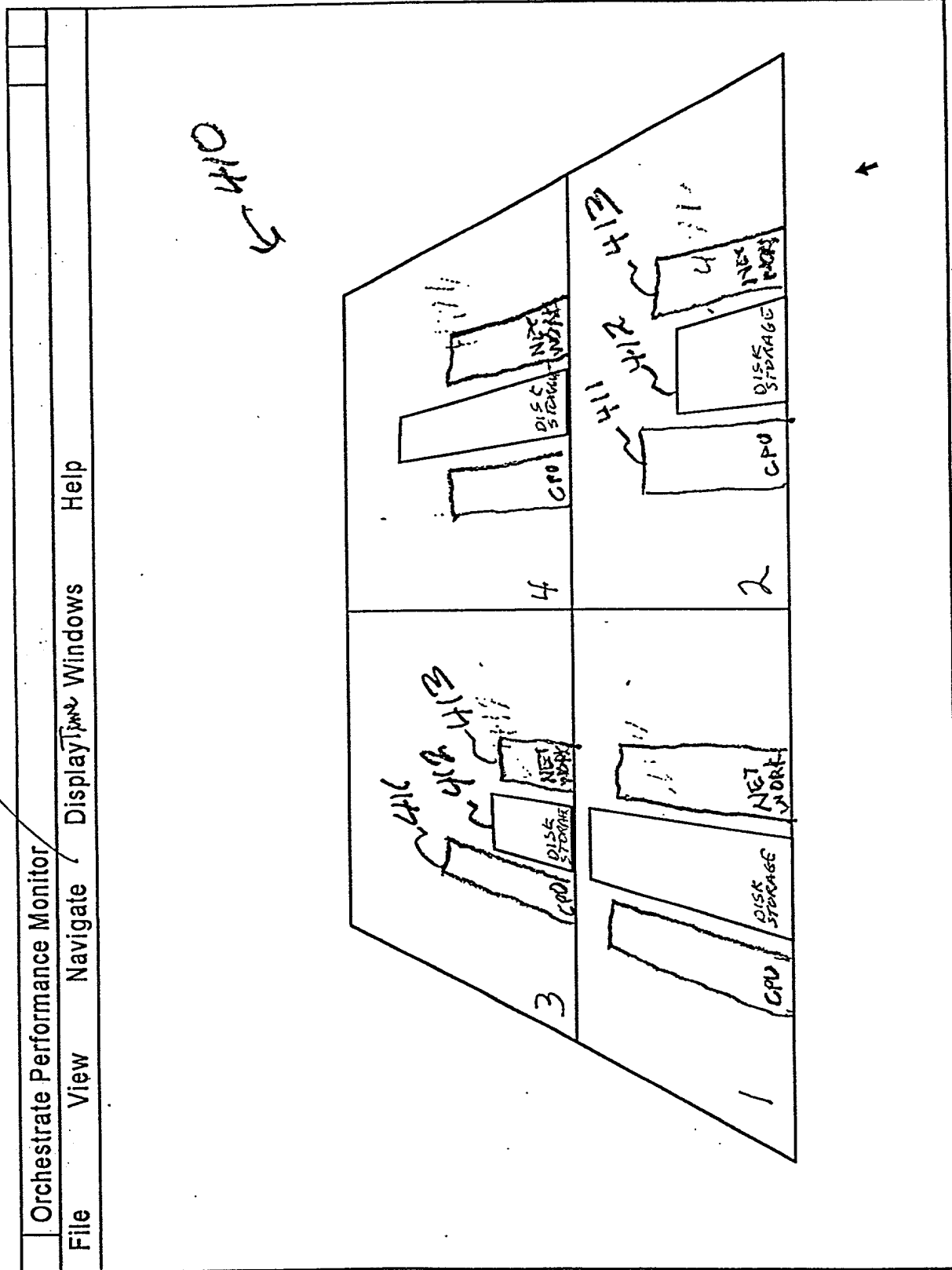


FIG. 1  
FIG. 2

10/10/2008 10:00:00 AM

Display

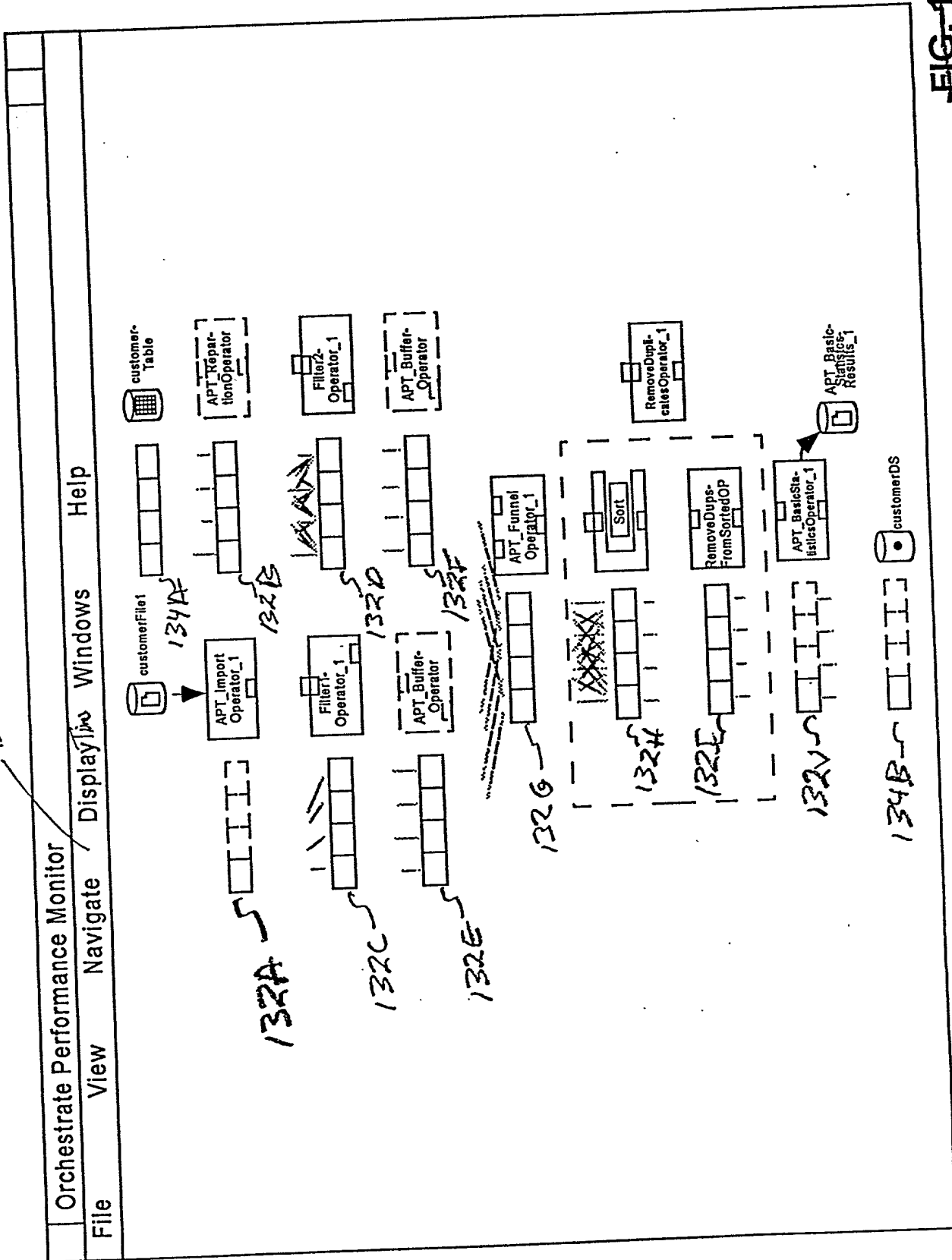


FIG-4  
F7615

-barGraphDisplayDialogBox3~238

-bar graphs on

-[ ] persistent data set~240

-[ ] buffer operator~241

-[ ] operator~242

- graph

-show~243

-flow rate in ( ) bytes, or ( ) record ~244

-[ ] outputs and/or [ ] inputs

-total flow to date in ( ) bytes, or ( ) records~245

-[ ] outputs and/or [ ] inputs

-[ ] relative to total records in input data set~245A

-disk I/O rate in ( ) bytes or ( ) records~246

( ) plain, ( ) % of maximum, or ( ) compared to remaining

-network I/O rate in ( ) bytes or ( ) records~248

( ) plain, ( ) % of maximum, or ( ) compared to remaining

-[ ] disk storage as in bytes ~250

( ) plain, ( ) % of maximum, or ( ) compared to remaining

-[ ] CPU usage as % of maximum~251

( ) plain, ( ) % of maximum, or ( ) compared to remaining

-direction~252

-( ) perpendicular to node array

-( ) on surface of each node

-[ ] label scale on graphs~258

-ok

-cancel

-help

16  
FIG. 14



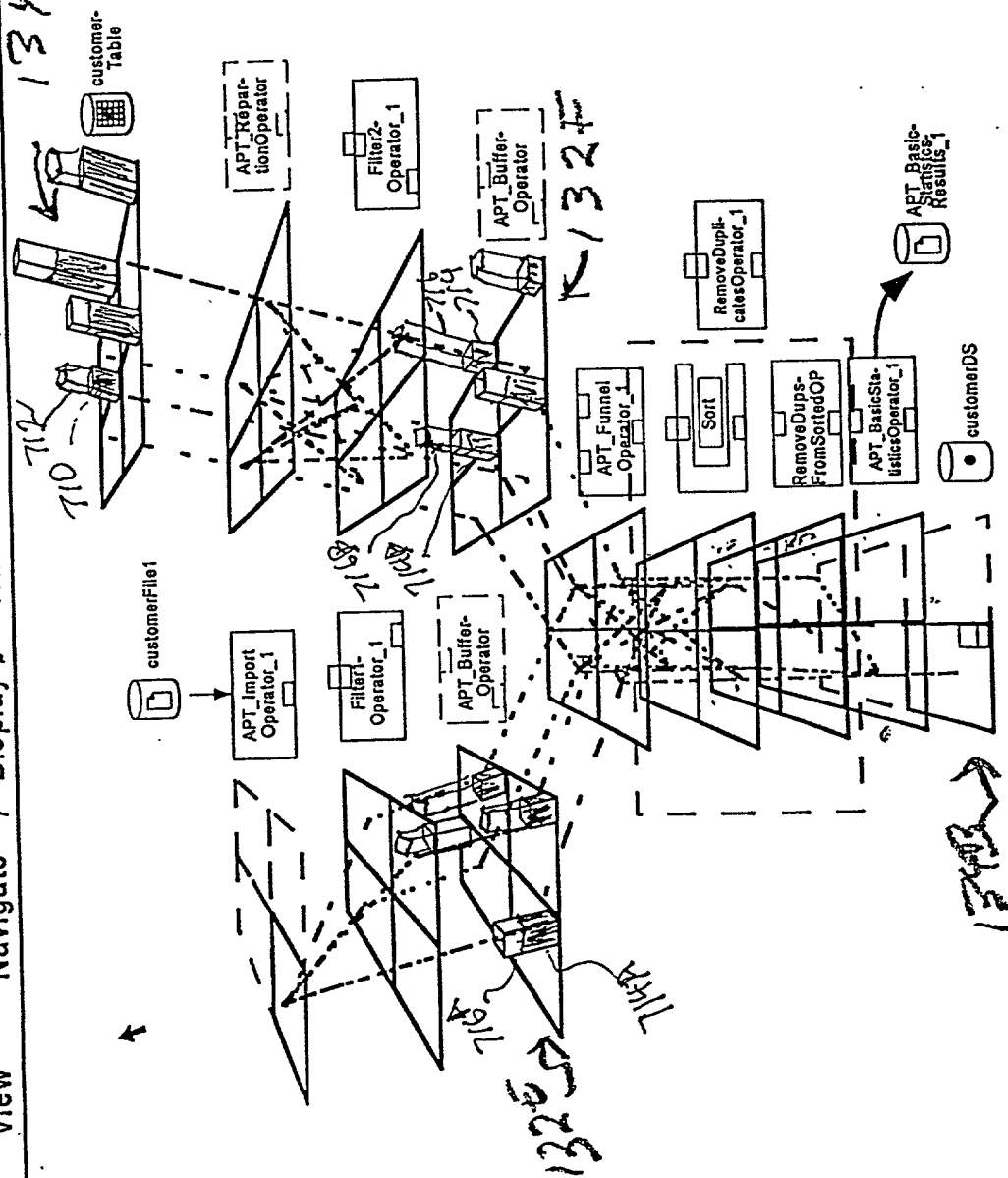


FIG. 1  
1917

TOO DATA COLLECTION

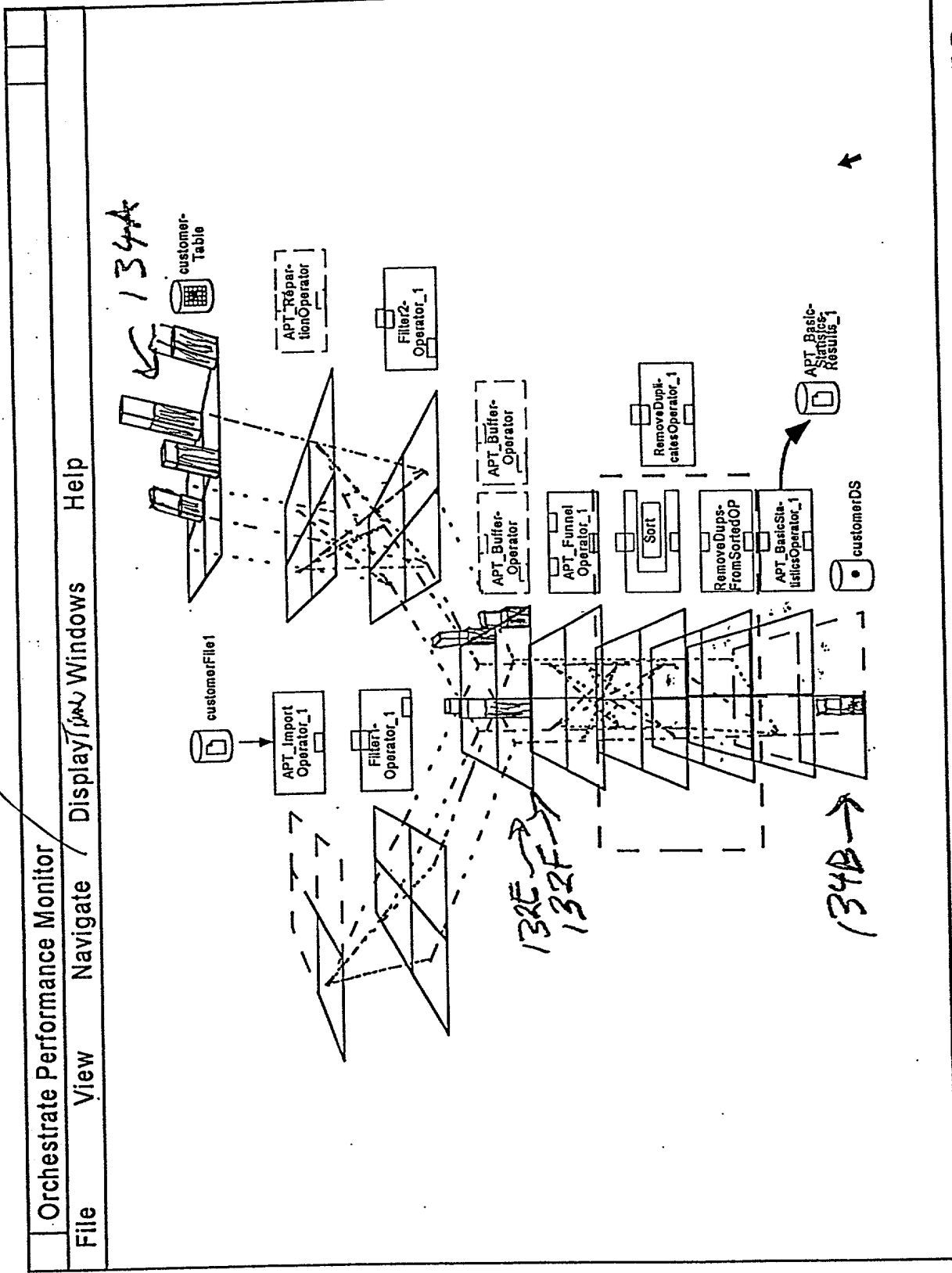


FIG-1  
8

*Data Collector*

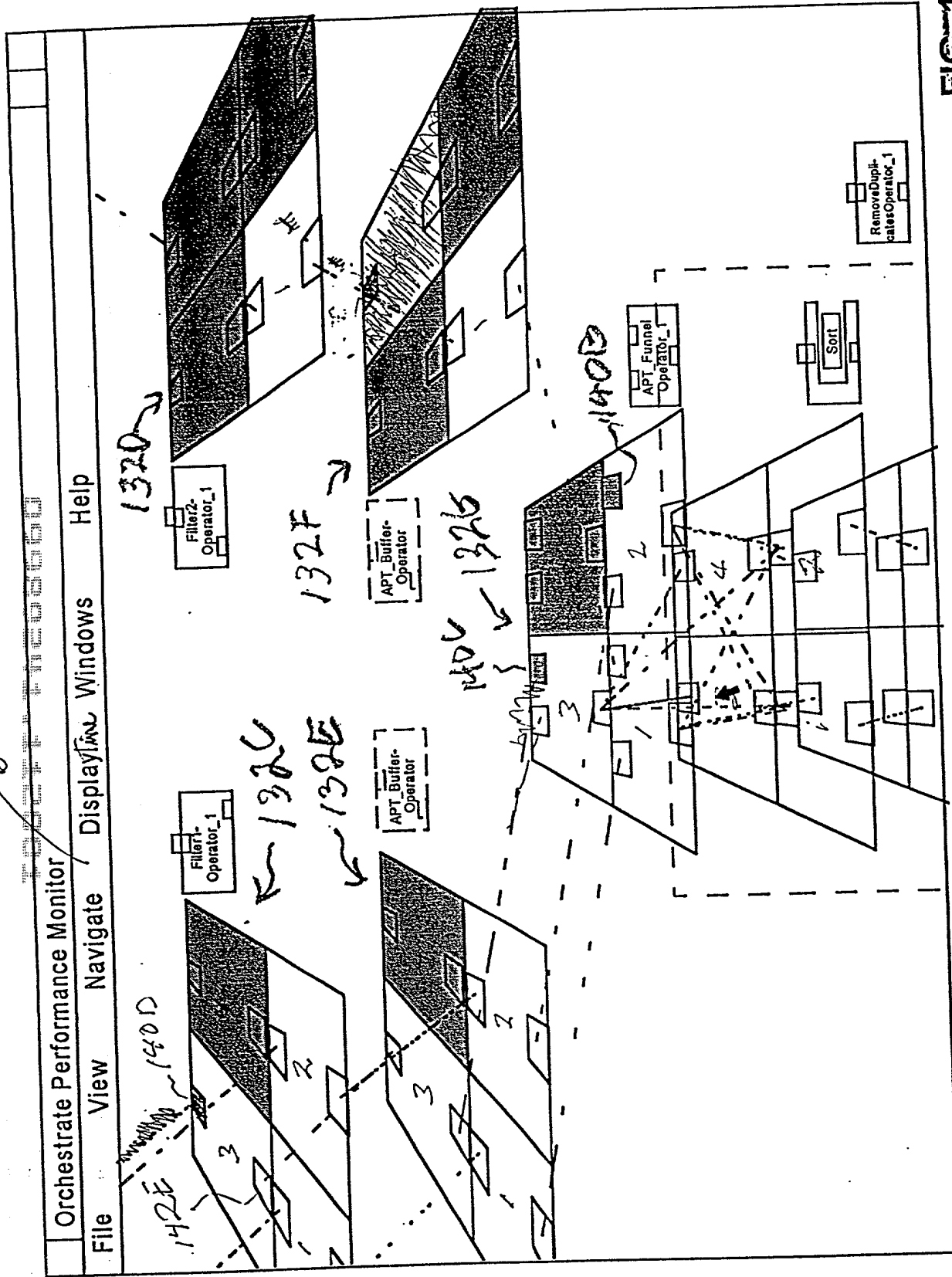


FIG. 1  
7/5/9

- SCANNED, #14
- datalinkDisplayDialogBox~260
    - [ ] display datalinks~261
    - color data link by~263
      - source ( ) node or ( ) operator~264A
      - destination ( ) node or ( ) operator~264B
      - flow rate in ( ) records or ( ) bytes~264C
      - inverse flow rate in ( ) records or ( ) bytes~264D
      - total records sent in ( ) records or ( ) bytes~264E
      - inverse of total records sent in ( ) records or ( ) bytes~264F
      - ( ) partition method~264G
      - monitored field value
        - by ( ) 1st letter, ( ) 1st number, or ( ) color map~264H
      - monitored field settings (button)~265
    - segmentation
      - ( ) solid~266A
      - data rate rain by
        - ( ) records/sec or ( ) bytes/sec~266B
      - ( ) color history ~266C
        - ( ) show entire time of graph execution along datalink~267A
        - ( ) show [editbox] secs along datalink~267B
        - separately color every [editbox] secs~267C
        - [ ] show time scale along datalinks (checkbox)~267D
    - color map dialog (button)~268
    - flow rate time frame~269
      - ( ) avg in last second
      - ( ) avg in last minute
      - ( ) avg in last [edit box] sec
    - ok
    - cancel
    - help

**FIG. 20**

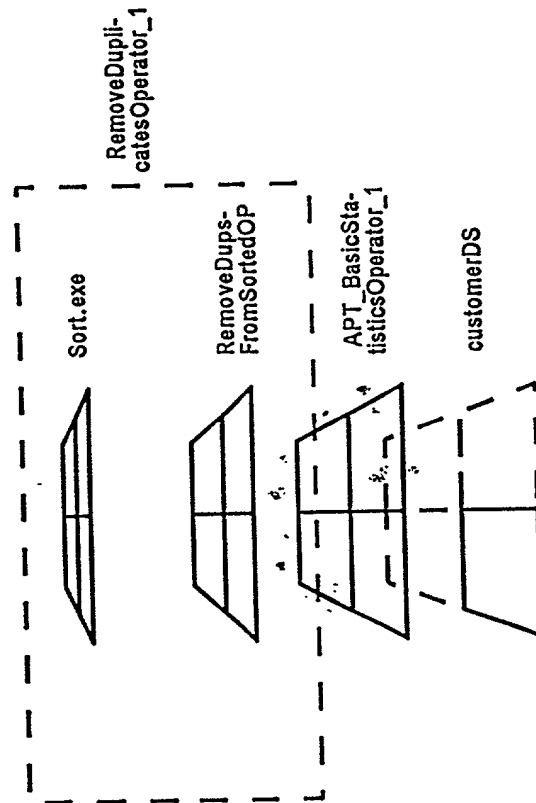
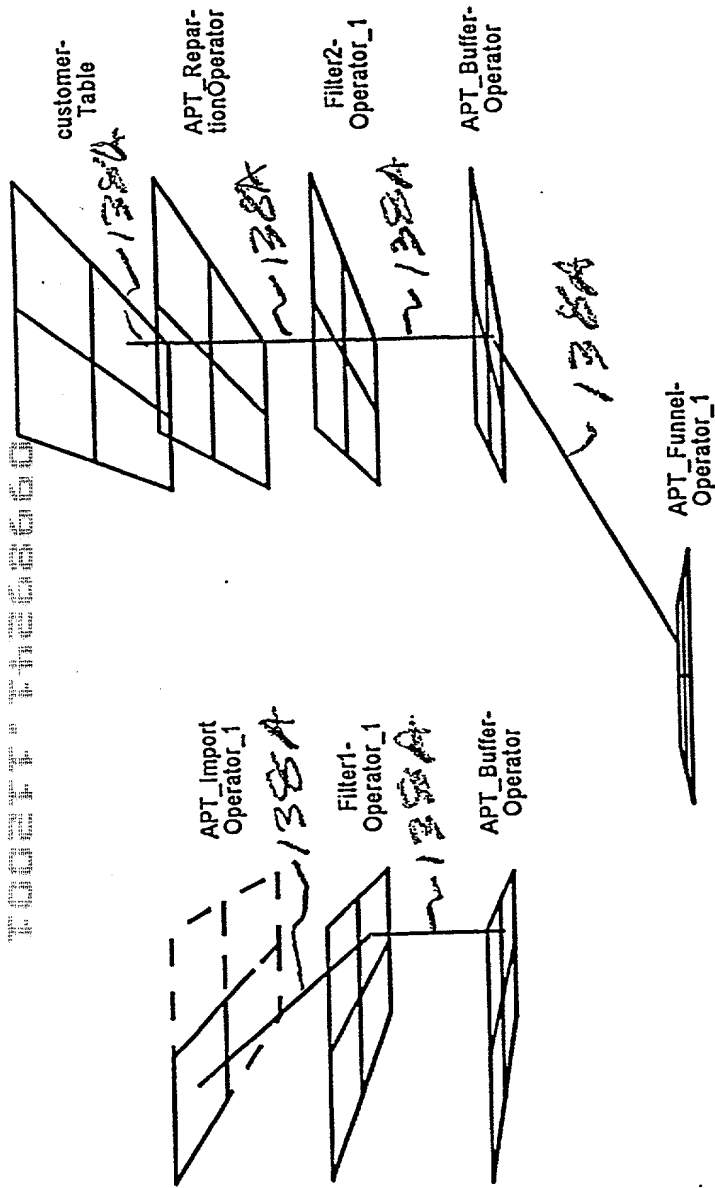
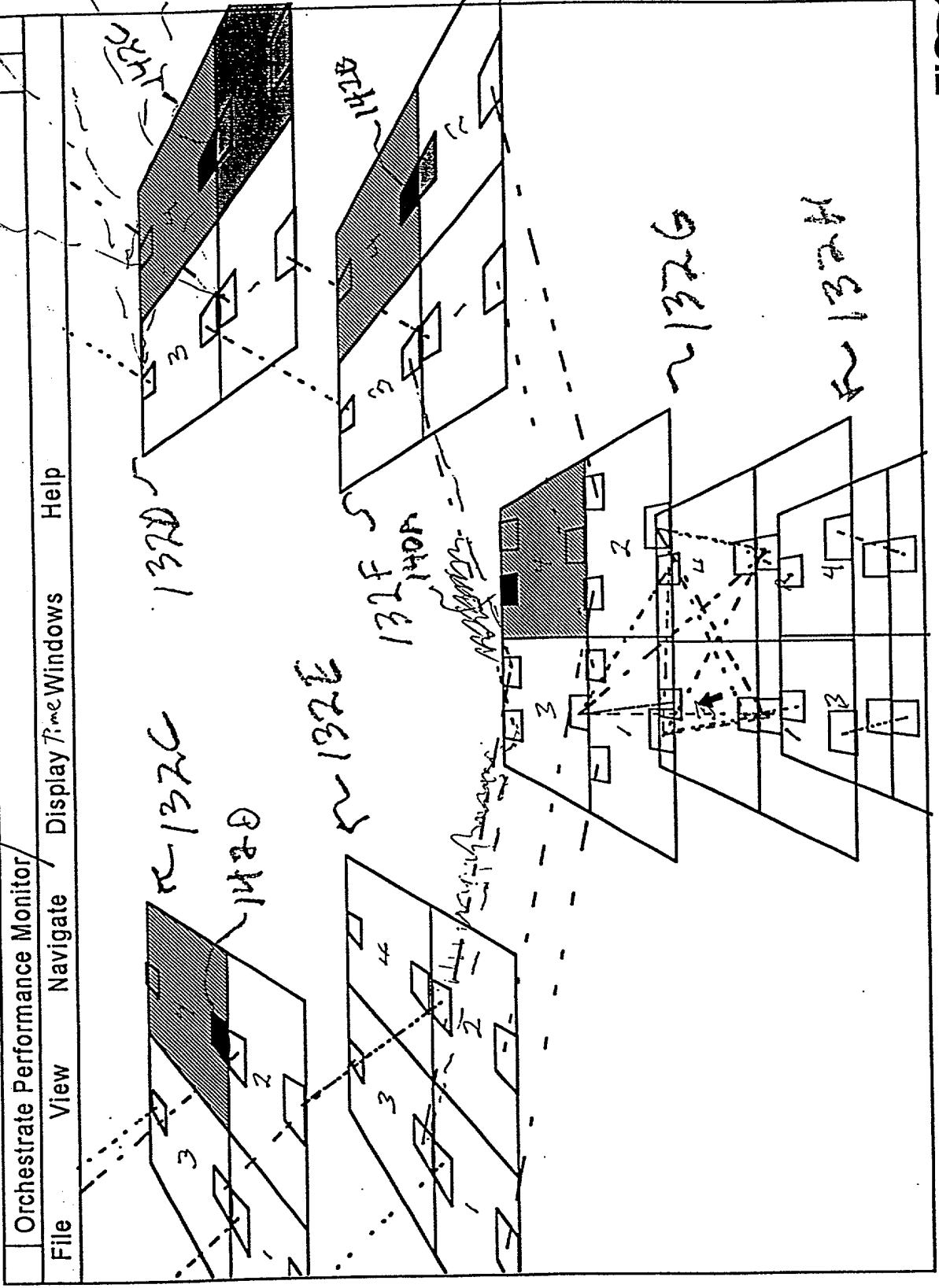


FIG. 1  
F/G

Data Collection

Orchestrate Performance Monitor



TO CHARTERED  
BY AIRLINE  
NOW

FIG. 1

FIG. 2A

374

378 361

375 376

Med ia

Monitor Field Dialog Box

☒ Field Monitoring On

Field Name  Monitoring Frequency  Every  Secs

☒ Field Display Window  Every block Sent Msg

☐ Trace by Desired Values 1st Desired Value  2nd Desired Value

Match

398 ☐ Initial Letters

400 ☐ All Letters

402 ☐ Msg First Occurrence

404 ☐ Balls ☐ Triangles

394

398 ☐ Initial Letters

400 ☐ All Letters

402 ☐ Msg First Occurrence

404 ☐ Balls ☐ Triangles

Ok Cancel Help

384  
382

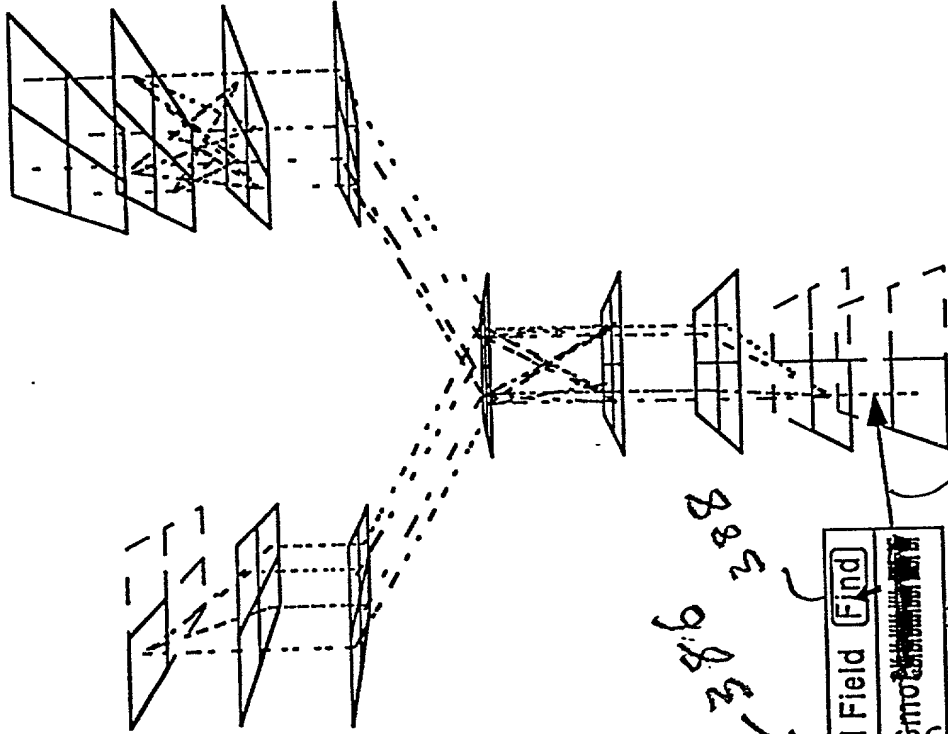
With Find Occurs 382

FIG. 24

FIG. 1  
FIG. 25

Orchestrate Performance Monitor

File View Navigate Display Windows Help



Monitored Field	Find
Last Name: Smith	

FIG. 1  
FIG. 25



TO: [unclear]  
 FROM: [unclear]  
 SUBJECT: [unclear]

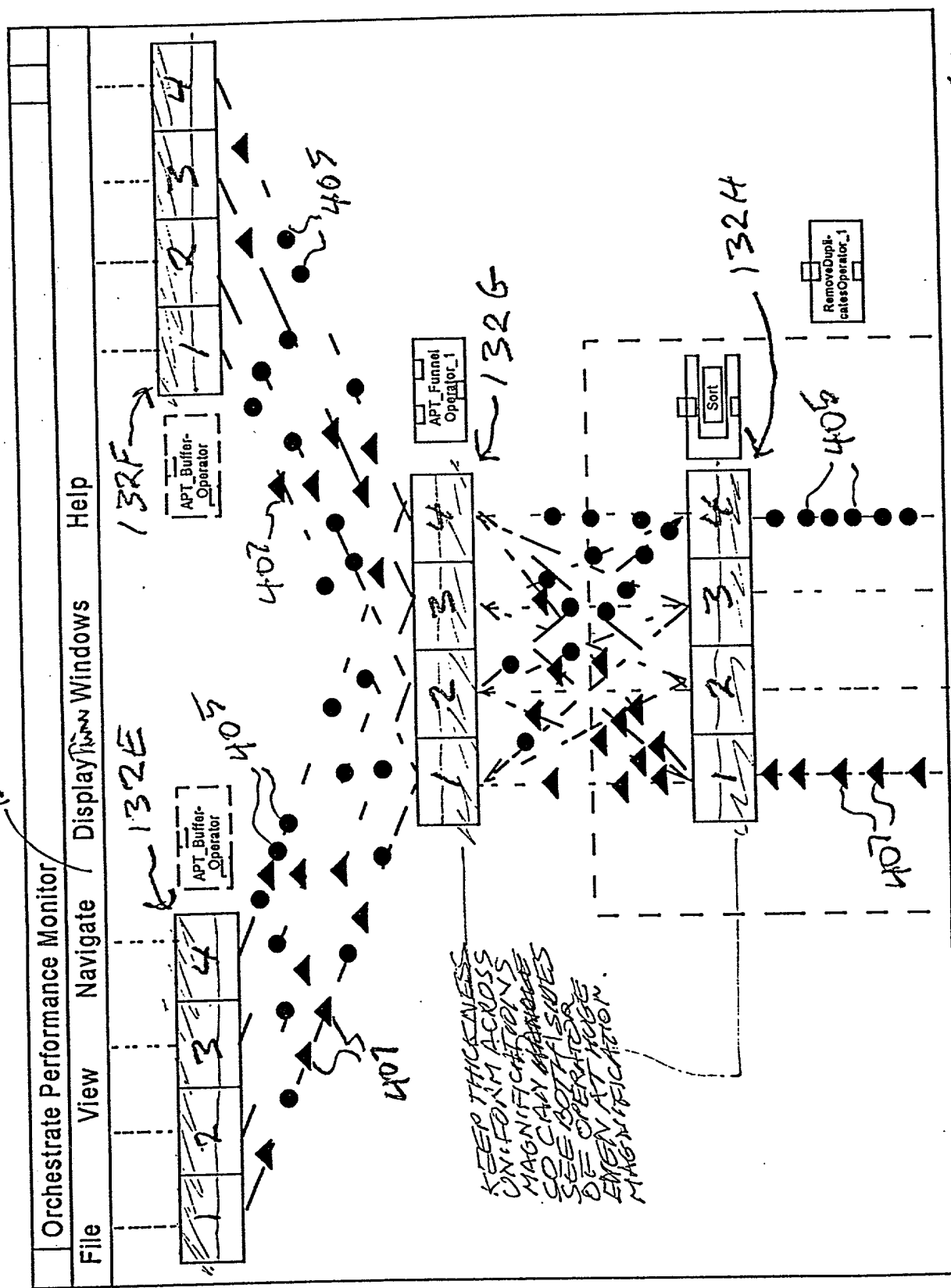


FIG 26

- getAndPutHangsDialogBox~266
  - [ ] monitor get and put hangs~268 271
  - consider call hung after [editbox] secs~270
  - highlight~272
    - [ ] port with hung call~274
    - [ ] hung operator instances~276
    - [ ] hung datalinks~278
  - [ ] don't highlight hangs for ports which have not handled any records~279
  - color hung getRecords as solid [color button grid]~280
  - color hung putRecords as solid [color button grid]~282
  - [ ] display time of hang~284
  - ok
  - cancel
  - help

**FIG. 23**

- EOFDisplayDialogBox~290
  - display EOFs by
    - [ ] port~292
    - [ ] operator~294
    - [ ] datalink~296
  - highlight EOFed objects as solid [color button grid]~298
  - [ ] display time of EOF~300
  - ok
  - cancel
  - help

**FIG. 27**

- DataCollection Menu~304
  - record performance~306
    - not at all~312
    - every second~314
    - every minute~316
    - custom~318
  - monitor performance~308
    - not at all~320
    - every second~322
    - every minute~324
    - custom~326

**FIG. 28**

- DisplayTime Menu~330
  - realtime~332
  - replay~334

**FIG. 29**

-replayDialogBox~336  
-time slider~338  
-replay speed slider~340  
-...  
-ok  
-cancel  
-help

FIG. 30

-operator objectMenu~342-343  
-make focus and nav~354  
-collapse instances~356  
-collapse operators at level~355  
-object overview~358  
-performance overview~360  
-datalink display~362  
-bar graph~368  
-monitored fields~370  
-...

FIG. 31

-operator port objectMenu~344  
-make focus and nav~354  
-collapse instances~356  
-object overview~358  
-performance overview~360  
-datalink display~362  
-bar graph~368  
-monitored fields~370  
-monitor records~372  
-...

FIG. 32

-operator instance objectMenu~346  
-make focus and nav~354  
-collapse instances~356  
-debug~357  
-object overview~358  
-performance overview~360  
-datalink display~362  
-bar graph~368  
-monitored fields~370  
-...

FIG. 33

- operator port instance objectMenu~348
  - make focus and nav~354
  - collapse instances~356
  - debug~357
  - object overview~358
  - performance overview~360
  - datalink display~362
  - bar graph~368
  - monitored fields~370
  - monitor records~372
  - ...

**FIG. 34**

- datalink objectMenu~350
  - make focus and nav~354
  - collapse instances~356
  - object overview~358
  - performance overview~360
  - datalink display~362
  - monitored fields~370
  - monitor records~372
  - ...

**FIG. 35**

- datalink instance objectMenu~352
  - make focus and nav~354
  - collapse instances~356
  - debug~357
  - object overview~358
  - performance overview~360
  - datalink display~362
  - monitored fields~370
  - monitor records~372
  - ...

**FIG. 36**

Added window.

Orchestrate Performance Monitor

File

View

Navigate

Display Windows

Help

Operator Instance Statistics Performance Overview

Find

3

2

4

3

2

4

-OperatorClass = APT\_Buffer-Operator

-Description: APT\_BufferOperators are automatically installed by Orchestrate to prevent data flow in locations in a graph where the graph's topology indicates such a blockage might be likely. It buffers records to disk if the input port of the operator they are intended for cannot currently receive them.

-Number Of Node Processor Is Running On = 4

--Percent of Node's CPU operator is currently using = 85%

-InputPort0

-PartitioningMethod Expected of Input = Same

-DataRate Over Last Second

= 0 rec/sec

= 0 byest/sec

-Total Data Received

= 1,969,528 records

= 1,008,398,336 bytes

-OutputPort0

-Hoisted PartitioningMethod= Same

-DataRate Over Last Second

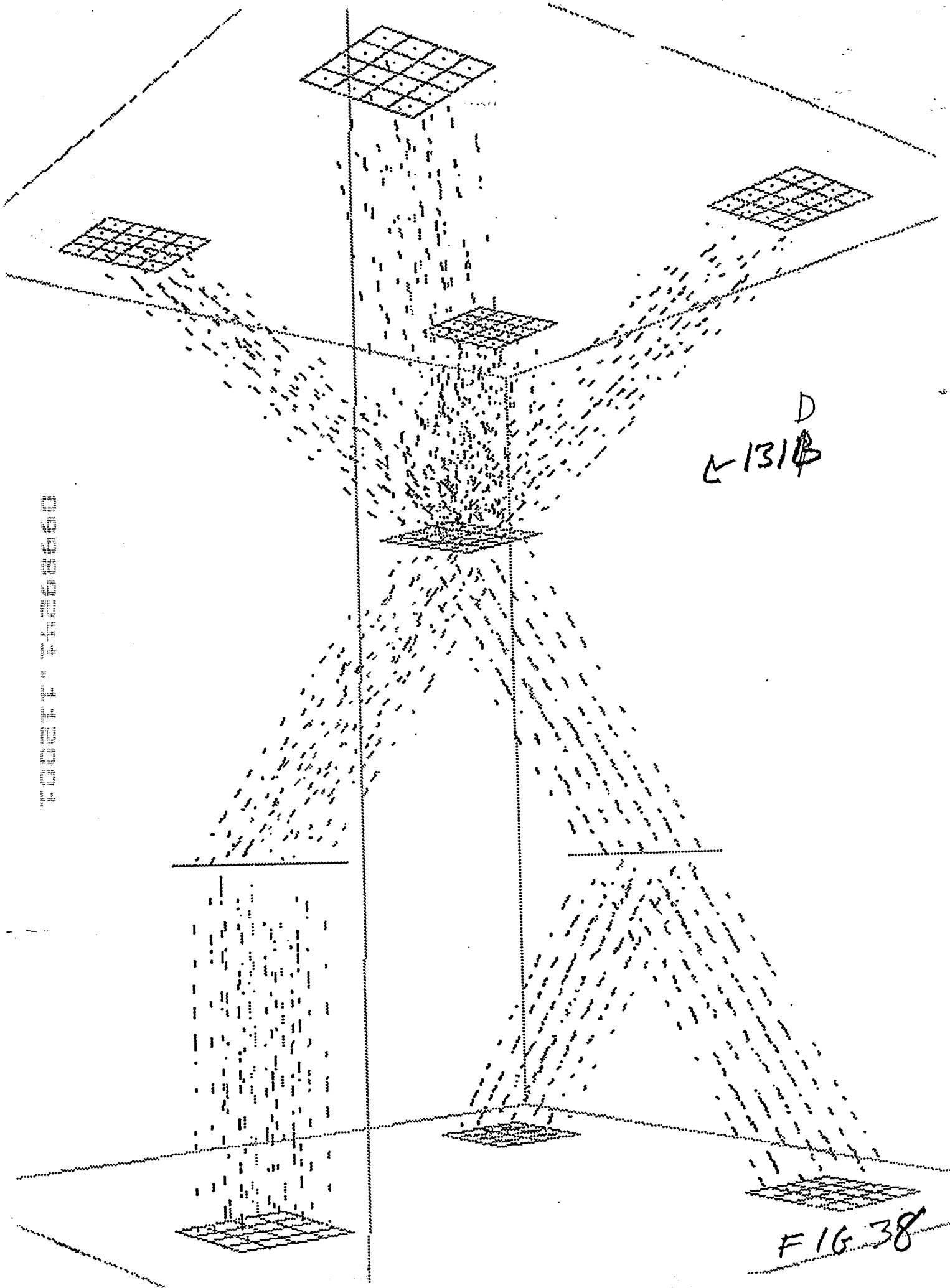
= 0 rec/sec

= 0 byest/sec

-Total Data Sent

FIG. 17  
FIG. 37

FOOTPRINT 140000



D  
← 131B

FIG. 38

FOOT THESSOO

L1316

FV 39

-putRecord~152

-place record in buffer block addressed to one of one or more instances of the output port's consuming input port based the partitioning scheme~420

-if record fills buffer block~422 *on*

-if getAndPutMonitoringIsOn, send the performance monitor timed putBlockPending UDP msg, identifying the sourceNode, sourceProcessOnNode, and sourceOutputOnProcess~424

-if output port's datalink is connected to a data set, write block to that data set~425

-else send buffer block to its corresponding consuming input port instance over TCP/IP~426

-call buildAndSendBlockSentMsg~423

-return~462

FIG. 40

-buildAndSendBlockSentMsg~429

-if current block contains EOF or if time since the last blockSent message was sent for current datalink exceeds desired blockSent interval~427

-assemble standard blockSent UDP msg including appropriate values for the message's sourceNode, sourceProcessOnNode, sourceOutputOnProcess, destinationNode, destinationProcessOnNode, destinationInputOnProcess, numberOfRecordsSoFar, numberOfBytesSoFar, timeSent, and isEOFInRecord~428

-If fieldMonitoringIsOn~430

-for each entry in monitoredFieldTable having matching source and destination~432

-if time since lastTimeRecordSent is greater than desiredSendInterval, place monitorFieldHeader in blockSent, followed by numberOfDesiredValue1, numberOfDesiredValue2, and the extracted field value from last record in current block~434

-If recordMonitoringIsOn~450

-if entry in monitoredRecordTable has matching source and destination~452

-if time since lastTimeRecordSent is greater than desiredSendInterval~454

-place monitorRecordHeader *as the* last record ~~X~~ in blockSent~456

-send performance monitor the blockSent UDP message~460

return~461

FIG. 41



-getRecord~148

- if there is no record in input port's input buffer~464
  - if getAndPutMonitoringIsOn, send performance monitor timed getBlockPending UDP msg, identifying the node, processOnNode, and inputOnProcess~465
  - if input port's datalink is connected to a data set~466
    - read next block from that data set~467
    - call buildAndSendBlockSentMsg~468
  - else wait until input buffer gets a new block of records~469
  - set inputBlockReadPointer to start of new block~470
- set return value to value of inputBlockReadPointer~474
- increment inputBlockReadPointer by length of record~476
- return~478

FIG. 42

-performanceMonitor~128

- initialize~480
- receive score description from process composing score of, and conducting the parallel execution of, the data flow graph~482
- create data structures for all nodes, operators, operator instances and their ports, and datalink instances~484
- determine which operators are in which levels~486
- lay out operators in each level in both 2 and 3-d space~490
- loop until user exits from within~492
  - while unprocessed user input, call handleUserMsgs~494
  - while msgs in UDP input queue, call handleUDPMsgs~496
  - for each graphState portrayed in a replay window, call UpdateReplayGraphState~532
  - if time since timeOfLastBlockPendingCheck exceeds blockPendingCheckInterval, call blockPendingCheck~550
  - for each graph visualization window~554
    - if it has been longer than windowUpdateInterval since it was last updated~569
      - if window is a 3Dview ~570
        - if a flight path is active for window, and if time since last move along flight path exceeds flight path's flightPathFrameInterval, change view parameters according to flight path~572
        - if its viewRadiusFromFocus, viewLongitudeFromFocus, viewLatitudeFromFocus, or viewZoom has been changed, call updateWindows3Dto2DProjection~574
        - call updateWindowsPixels~576
      - if the window is a 2D view update it~629 621

FIG. 43

-handleUDPMsgs~498  
 -for each message in the queue~500  
   -if getBlockPending msg, store it in data structure for its associated input port instance~502  
   -else if putBlockPending msg, store in data structure for its association<sup>ed</sup> output port instance~504  
   -else if blockSent msg~506  
     -if historyRecordingIsOn~508,  
       -if time since last blockSent was recorded for the msg's data link instance exceeds blockSentRecordInterval, store a copy of blockSent msg in the blockSentHistory file in chronological order by time stamp~510  
       -if blockSent's isEOFInRecord is true, record blockSent in EOFHistory file~511  
   -call updateGraphWithBlockSentMsg with the blockSent msg for realTime graphState~512  
 -return~530

FIG. 44

-updateGraphStateWithBlockSentMsg~ 531  
 -erase any getBlockPending, putBlockPending, hungOnGetRecord, hungOnPutRecord in data structures for all inputs or outputs on same instance having an earlier time stamp than the blockSent msg~513  
 -if there is a prior blockSent msg recorded for the data link, calculate dataRates for the current blockSent msg's corresponding datalink instance by subtracting numberOfRecordsSoFar and numberOfBytesSoFar from the prior blockSent stored in data structure of the datalink instance having msg's associated source and destination ports from numberOfRecordsSoFar and number of BytesSoFar, respectively, in the current msg and divide difference by the difference in timeSent of the two msgs~514  
 -store the dataRate in the datalink instances data structure~515  
 -store the current value of blockSent in the datalink instance's data structure~516  
 -if blockSent's isEOFInRecord is true, set hasSentEOF in blockSent's corresponding datalink instance~517  
 -if there is a monitoredFieldHeader in the current blockSent~522  
   -if there is a window opened to display monitored field information for the datalink instance, feed the monitored field information to that window~524  
   -if msgFirstOccurrence has been set for either the 1stDesiredValue or 2ndDesiredValue, check if the blockSent msg contains the first occurrence of the either desired value and if so beep and send appropriate window to display~526  
 -if blockSent contains a monitoredRecordHeader and if there is a window opened to display that record for the datalink instance, feed blockSent's monitored record information to that window's process~528  
 -return~529

FIG. 45

-updateReplayGraphState~534

-if the time since lastGraphStateUpdateTime exceeds replayUpdateInterval~535

-save the current time as lastRealTimeOfGraphStateUpdate~536

-find next timeBeingReplayed ~537

-scan EOFHistory file and get&PutHangHistory files since the graphState's previous timeBeingReplayed, to update EOF and getRecord and putRecord hang values of port and operator instances~538

-clear the blockSent msgs and dataRate stored for all datalinks in the graphState~539

-for each item in blockSentHistory file recorded from timeBeingReplayed back to timeBeingReplayed - replayUpdateWindowDuration~540

-if the blockSent msg corresponding datalink for which a dataRate has not been calculated, in the GraphState call updateGraphStateWithBlockSentMsg~542

-return~543

## FIG. 46

-blockPendingCheck~552

-for the data structure of each port instance~556

-if it has a getBlockPending or putBlockPending indication with a time stamp older than normalBlockPendingTime~558

-label the data structure for port as hungOnGetRecord or hungOnPutRecord, respectively~560

-if recording is on, record hungOnGetRecord or hungOnPutRecord, respectively, and its associated time for port instance in get&PutHangHistory file~562

-Return~564

## FIG. 47

-updateWindows3DTo2DProjection~578

-set viewDirectionLongitude and viewDirectionLatitude to keep view centered on viewFocus~582

-call calculateGraphs2DProjectionCoordinates~584

-return~586

## FIG. 48

-updateWindowsPixels~588

- for all datalink objects in the window which have to be updated and which might contain any transparent segments, erase all pels associated with it~589

- for each object in the visualization window's graphState mapped into the window's 2D projection which has been changed since the last call to this routine~590

- if it is a datalink object~592

- if datalink object's colorSetting is "off", skip rest of loop~594

- else if datalink object's colorSetting is a function of a selected variable~596

- if object is a composite, set its value for the selected variable to the average of the values of that variable from all its corresponding instances~598

- select object's draw color from the position of its value for the selected variable relative to the variable's color map~600

- else if datalink object's colorSetting is a fixed color set its draw color equal to that color~602

- else if datalink object's segmentationSetting is for a color histogram, call colorHistogramSegmentation~604

- if datalink object's segmentationSetting is for dataRate noodles, call noodleSegmentation~606

- if datalink object's segmentationSetting is Solid, set lineSegmentBuffer to one solid segment~608

- call drawLineSegments~610

- if it is not a datalink object, project it into the view's pel map, taking into account what objects block what other objects from perspective of viewpoint~614

...

- return~616

FIG. 49

-colorHistogramSegmentation~618

- if time or time scale represented by window has changed, update outputPortPointer and inputPortPointer in the lineSegmentBuffer accordingly~620
- if any part of the object's lineSegmentBuffer between its outputPortPointer and inputPortPointer is not segmented, calculate histogram color segmentation for that portion of the buffer from either real time values, or if appropriate, relevant portions of blockSentHistory file~622
- return~624

FIG. 50

-noodleSegmentation~626

- advance outputPortPointer and inputPortPointers in lineSegmentBuffer in proportion to current dataRate~628
- for each part of the object's lineSegmentBuffer between its OutputPortPointer and inputPortPointer which is empty, starting at the outputPortPointer~629
  - until such empty portion has been filled~630
    - add a new pair of "on" and "off" segments to the buffer, having a density which is a partially random function of link's dataRate for the time frame represented by that segment's location along the line representing the datalink object~631
    - if monitoredFieldDisplaysOn, if the datalink object has a monitored field, and if a blockSent message is detected in the time represented by such pair has a value for that field represented by either a triangle or a ball, replace the pair's "on" segment in the buffer with a triangle or ball~634
- return~635

FIG. 51

-drawLineSegments~636

- Erase all pixels, if any, previously associated with the current data object by step 648 in a previous call to the function routine ~ 63*
- for each line segment in the current datalink object's lineSegmentationBuffer which maps into the window~638
  - if its color is "off" skip rest of loop~642
  - for each successive portion of that line segment which has been projected onto an individual pel of the view~643
    - if pel was not formerly colored~644
      - if the segment's color is "on", color the pel with the datalink object's draw color~645
      - else color the pel with the segment's color~646
      - store the ID of the current datalink object in association with the pel~648
    - else, if the current datalink object is between the view point and the object previously associated with the pel at the point on each object represented by the pel, store the color of the current datalink object at the pel and associate the current datalink object with the pel~650
- return~652

FIG. 52